УДК 004.89; 004.416

Е.А. Шалфеева
Институт автоматики и процессов управления ДВО РАН, Россия
Россия, 690041, г. Владивосток, ул. Радио, 5

Метод построения проектных представлений интеллектуального решателя задач по моделям начальных стадий жизненного цикла*

E.A. Shalfeeva
The Institute of Automation and Control Processes, Vladivostok, Russia
Russia, 690041, Vladivostok, 5 Radio Street

The Method of Construction of Intelligent Problem Solver's Design Representations from the Models of Life Cycle Initial Stages

О.А. Шалфєєва
Інститут автоматики і процесів управління ДВО РАН, Росія
Росія, 69041, м. Владивосток, вул. Радіо, 5

Метод побудови проектних представлень інтелектуального вирішувача задач на моделях навчальних стадій життєвого циклу

В статье представлен метод построения решателя задач интеллектуальной программной системы путем преобразования ранних ее моделей в его проектные модели, представления которых удобны и для человека и для программной обработки. Базированные на этом методе технология разработки и средства поддержки разработчиков обеспечат сопровождаемость такой системы.
Ключевые слова: решатель интеллектуальной задачи, иерархическая семантическая сеть, сопровождение интеллектуальной программной системы.

In the article the method of construction of a solver of problems of intelligent software by transformation of its early models to its design models is presented. The representations of all these models are convenient both for the person and for program processing. Based on this method intelligent software engineering and supporting tools will make for a maintenance of such system.
Key words: authentication, intelligent problem solver, hierarchical semantic network, intelligent software maintenance.

У статті представлений метод побудови вирішувача задач інтелектуальної програмної системи шляхом перетворення ранніх її моделей на його проектні методи, представлення яких зручні і для людини, і для програмної обробки. Технологія розробки та засоби підтримки розробників, що базуються на даному методі, забезпечать супровідність даної системи.
Ключові слова: вирішувач інтелектуального завдання, ієрархічна семантична мережа, супровід інтелектуальної програмної системи.

---

*Работа выполнена при финансовой поддержке гранта РФФИ 12-07-00179-а и ДВО РАН № 12-I-ОНИТ-04.

Введение

Автоматизация профессиональной деятельности, в том числе интеллектуальной, является распространенной сферой приложения усилий IT-специалистов. Внедряемые взаимосвязанные комплексы программных, информационных и организационных средств (например, интегрированный программный комплекс перинатального центра [1]) оптимизируют документооборот, формируют отчеты, ведут учет расходов, протоколируют работу персонала. Однако такие комплексы не имеют внутри себя средств поддержки принятия решений специалистов (в частности, пользователи перинатального центра нуждаются в системе поддержки принятия врачебных решений [1]). Успехи в автоматизации поддержки принятия решений при решении интеллектуальных задач связаны с созданием систем, основанных на знаниях. Но в доступных литературных источниках не представлены примеры таких систем, которые бы эксплуатировались и развивались в реальных условиях продолжительное время.

В настоящее время от автоматизации интеллектуальной программной деятельности ожидают создания интегрированного интернет-комплекса средств поддержки принятия решения для требуемых задач и средств доступа к общим знаниям и другой информации. Интегрированный комплекс должен быть наращиваемым и сопровождаемым [2]. Это требует объединения идей из области искусственного интеллекта с идеями из области современного программирования [3-5].

В известных трех *фазах – определения, разработки, сопровождения* программного обеспечения (ПО) [6] – успех последней зависит от результатов двух предшествующих.

В фазе определения особая роль принадлежит деятельностям *системного анализа* (СА), от которых зависит реализуемость всей ИПС. Фаза определения дает множество моделей, от полноты которых зависит удовлетворенность пользователей ИПС, а фаза разработки – те, от которых зависит качество реализации и возможность сопровождения. В работе [2] показана возможность проектирования долгоживущей интеллектуальной программной системы (ИПС) на основе гипотезы о декларативном программировании.

Цель представляемого исследования: разработка такого метода преобразования ранних моделей решателя интеллектуальной задачи в его декларативные проектные представления, на основе которого возможна автоматизация разработки долгоживущих систем, основанных на знаниях.

Модели начальных стадий жизненного цикла, используемые при построении модели анализа требований

Полный системный анализ, выполняемый в рамках автоматизации профессиональной деятельности, завершается построением множества моделей ИПС и ее компонентов, обеспечивающих ее сопровождаемость [7], среди них – онтология знаний предметной области, концептуальная архитектура автоматизирующей системы, математическая постановка каждой задачи, алгоритм решения.

До формулирования задачи должны быть определены: термины действительности – «ненаблюдаемые неизвестные» (например, для медицинской диагностики – «диагноз», «интервалы развития признака» [8], [9]), ограничения целостности ситуаций действительности (например, «Интервал, на котором наблюдается развитие некоторого признака, покрывает все моменты наблюдения этого признака» [8]), соглашения о связях действительности и знаний («Если заболевание входит в диагноз, то число

периодов развития этого заболевания у пациента совпадает с числом периодов его развития в базе знаний, а длительность каждого периода развития лежит между нижней и верхней границами длительностями этого периода развития заболевания»).

Постановка задачи содержит метод решения задачи. Пример метода диагностики в медицине – «перебор всех возможных значений выходных данных (диагнозов – отдельных заболеваний); для каждого заболевания выполняется построение всех возможных вариантов развития причинно-следственных связей (на основе информации из базы знаний, значений анатомо-физиологических особенностей пациента и значений произошедших с ним событий) и поиск среди них всех тех, которым соответствуют все наблюдаемые значения признаков пациента» [9].

На основе «метода решения задачи» и «связей знаний с действительностью» (части онтологии знаний предметной области) аналитик или инженер знаний строит алгоритм решения (обычно онтологическим соглашениям о связях действительности и знаний сопоставляются отдельные подзадачи проверки выполнения соглашения).

Пример. Алгоритм, установленный математиком или аналитиком для «постановки и объяснения диагноза» [9], в виде разбиения на подзадачи с учетом их вложенности, выглядит так:

- Получить данные наблюдений пациента
- Проверить гипотезу о том, что пациент здоров
-- организовать Цикл по наблюдавшимся признакам
--- проверить гипотезу о том, что все наблюдавшиеся значения выбранного признака могут иметь место у здорового пациента.
--- запомнить значения признака, опровергающие гипотезу
-- подытожить гипотезу о том, что пациент здоров;
- организовать Цикл по заболеваниям
-- проверить выполнение необходимого условия для текущего заболевания;
-- проверить гипотезу о заболевании
--- попытаться отвергнуть гипотезу о заболевании по «не-КК»-признаку
--- организовать цикл по наблюдавшимся КК-признакам
---- проверить возможность наблюдавшихся значений
---- организовать цикл по вариантам проявления выбранного КК-признака
----- сопоставить динамику знаний варианту причинно-следственной связи (ПСС)
----- добавить анализ этого варианта в объяснение
---- подытожить результаты анализа всех вариантов в объяснении КК-признака
-- подытожить результаты анализа всех признаков в объяснении заболевания;
- выдать результат – множество диагнозов и их объяснение.

Примечание. КК-*признаком* назван здесь признак, входящий в клиническую картину заболевания, не-КК – соответственно – не входящий в нее.

Модели требуемой функциональности решателя

Анализ функциональности, поведения и обрабатываемой информации каждой подсистемы ИПС (в частности, специализированных решателей задач) тоже относится к фазе определения. Анализ требований, детализирующий функции, информацию и поведение подсистемы, проводится перед началом проектных работ (designing) каждой подсистемы ИПС. Для систем, основанных на декларативно-представленных знаниях,

наиболее важны первые две группы представлений. Обе они базируются на ранних моделях: функциональная модель – на математической постановке задачи и алгоритме, а модель обрабатываемой информации – на онтологии действительности.

Модель функционального разбиения для решателя отдельной задачи зависит в большей мере от двух ранних моделей – предполагаемого алгоритма решения и пользовательских требований к решателю. Обе, как правило, существуют к концу СА. Наиболее традиционным представлением для алгоритма является блок-схема. Пользовательские функциональные требования могут быть выражены графовой моделью «вариантов использования», либо текстовым описанием пожеланий потенциального пользователя.

Универсальным представлением для алгоритма решения задачи и модели функционального разбиения задачи [6] может считаться иерархия подзадач.

Модель разбиения на функции. Подзадача некоторого уровня (последовательная или с-ветвлением) разбивается на несколько различных мелких подзадач, к некоторым из них может быть необходимо «цикличное обращение». Возможно и более детальное описание отношений между подзадачами, например, вместо «последовательная» – «строгий порядок» или «произвольный», вместо «с-ветвлением» – «по выбору пользователя» или «выбор в зависимости от вычисленных значений». Кроме того, подзадача может быть помечена как «необязательная», а нетерминальная подзадача может быть «абстрактной» [10]. Свойство отношения между подзадачами соседних уровней «цикличное обращение» может быть помечено как уточнение: «Обращается циклично по элементам конечного множества», – или – «итеративное обращение пользователя» [10].

Иерархическую модель подзадач (ИМпЗ) в виде дерева разбиения естественно сохранять как *иерархическую семантическую сеть* (ИСС) – сеть с возможными полустепенями захода больше единицы.

В процессе разбиения для элементов иерархии (подзадач) очередного уровня важно различать, состоят ли подзадачи в «простом» подчинении, в «цикличном» подчинении, будут «выбираться» или «необязательны». Структура ИСС может быть такой (здесь альтернативные представления элементов сети и множества элементов помечены спецификаторами *alt* и *set*):

задача {
название: строка;
структурность: ~alt (последовательная; с-вариантами)
~set подзадача
{подчиненность: ~alt (в «простом» подчинении;
в «цикличном» подчинении;
«выбираемая по условию»)
название: строка;
структурность: ~alt (последовательная; с-вариантами)
[~set {–> подзадача}]
}}.

Ожидаемая автоматизация поддержки разработчиков на этом этапе такова. Поскольку артефакт создается на основе моделей предметной области, задач и создаваемой системы, то средства редактирования должны обеспечивать «контролируемое использование» содержимого этих артефактов: дать возможность просматривать, копировать значения и помечать использованные элементы предшествующих моделей.

Доступная для просмотра и копирования фрагментов онтология задачи может быть доступна как множество предложений с возможностью размечать их {не рассмотрено, использовано, не понадобилось} для последующего контроля полноты и прослеживаемости процесса разработки. Соответствующее предложение (соглашение о связях) после сопоставления ему подзадачи может помечаться как «использовано».

Для разработчика желательна такая поддержка: автоматическое создание артефакта «иерархия подзадач», как сеть с одной корневой вершиной-задачей с названием, совпадающим с названием в постановке задачи.

Аналитик нуждается в средствах сохранения своей модели, поскольку эта модель станет «ключевой» (по ней будут производиться последующие, более «технические», представления решателя подсистемы).

Модели связи функций решателя с данными. Традиционно анализ требований представляет связи функций с получаемыми и производимыми элементами информации. Для учета всех обрабатываемых данных подсистемы в модели каждое входное (или модифицируемое) данное следует различать как хранимое или интерактивно получаемое от пользователя, либо промежуточное, формируемое в процессе решения «внутри» решателя и не требующее сохранения. Содержание и структура хранимых и получаемых от пользователя данных обычно являются частью онтологии действительности, построенной на СА.

Расширенная модель подзадач (РМпЗ) обогащает ИМпЗ связями с данными, такими как «хранилища», в классической модели потоков данных [6].

Достаточная метаинформация для представления этой модели такова:

задача {
название: строка;
структурность: ~alt (последовательная; OR с-вариантами)
~set подзадача
{подчиненность: ~alt (в «простом» подчинении,
в «цикличном» подчинении,
«выбираемая по условию»);
название: строка;
структурность: ~alt (последовательная OR с-вариантами)
~set входное данное{~alt хранимое,
интерактивное,
внутри решателя ~set {-> подзадача}};
~set модифицируемое данное{~alt хранимое;
интерактивное;
внутри решателя ~set {-> подзадача}};
[~set {–> подзадача}]}};

Пример представления некоторой задачи из «РМпЗ» в виде сети:

подзадача-2-1-1 [подзадача]
Оценка гипотезы «признак в норме» [название];
в «цикличном» подчинении [подчиненность];
признак [параметр цикла]
Дневник наблюдений. запись [входное данное]
хранимое
БЗ о Норме. признак [входное данное]

хранимое
объяснение. гипотеза Здоров. опровергающий гипотезу признак
[модифицируемое данное]
хранимое
подзадача-2-1-2 [подзадача].

Ожидаемая автоматизация поддержки разработчиков на этом этапе такова.

Автоматизированная поддержка создания такого артефакта состоит в генерации узлов-подзадач со значениями атрибутов «название, структурность и подчиненность», скопированными из ИМпЗ, и возможности порождения необходимого количества элементов сети «входное данное» и «модифицируемое данное» для каждого узла-подзадачи.

Построение архитектурных представлений решателя

Фаза разработки включает в себя создание архитектурных представлений каждой подсистемы (специализированного решателя), создание прочих проектных представлений всех составных «частей» ИПС и собственно реализацию всех проектных решений. От архитектурных моделей зависит качество реализации и возможность сопровождения [6], [11].

В процессе архитектурного проектирования относительно независимой подсистемы (такой как решатель ИПС) акцентируют внимание, во-первых, на модульной декомпозиции и распределении функциональности между этими модулями (units, «единицами»), а во-вторых, на моделировании управления – определении взаимодействия между ними [6].

Программными единицами (ПЕ), в зависимости от подхода к реализации, могут становиться процедурные единицы, объекты/классы, агенты или другие единицы. Построение специализированного решателя с помощью архитектуры вызова-возврата [6] (когда каждая ПЕ обращается к некоторой другой ПЕ – функции или процедуре, или операции класса объектов – путем ее вызова с ожиданием возврата управления) может быть осуществлено известными методами [6].

В этом случае необходимо дополнительно обеспечить возможность обращаться из ПЕ к элементам ИСС, хранящим знания, данные и формируемое объяснение.

Другой подход – построение специализированного решателя, работающего со знаниями и данными в виде ИСС, в архитектуре с двумя типами ПЕ. Один тип – активные, относительно самостоятельные активные единицы, сообщающиеся друг с другом посредством сообщений (назовем их агентами [4]), второй тип – операции над классами семантических сетей (по аналогии с операциями над АТД – абстрактными типами данных).

Для такой архитектуры в соответствии с моделями задач определяется сначала модель необходимых ПЕ-агентов, затем моделируются взаимодействия ПЕ друг с другом, далее дополняется необходимыми ПЕ-операциями.

Модель всех требуемых ПЕ. В процессе распределения функциональности, представленной в ИМпЗ, по основным ПЕ принимается решение по количеству единиц, а связь по управлению только «намечается», т.е. указывается, как единицы друг другу «делегируют часть работы»: агент привлекает другого агента к сотрудничеству.

Агентов (или блоков агентов) должно быть предложено столько, чтобы скоординировать и выполнить имеющиеся подфункции.

В случае выбора вышеобозначенного «агентного подхода» метатеория модели всех требуемых ПЕ, представляемой «иерархически», такова:

сеть агентов {
Корневой агент [строка]
~ set сотрудничество {имя агента-сотрудника [строка]{
тип агента [управл, обрабат, группирующий]
~ setmm -> сотрудничество}}};

Такая сеть агентов показывает, в «услугах» какой ПЕ нуждается каждая ПЕ решателя.

Ожидаемая автоматизация поддержки разработчиков на этом этапе такова. Процесс распределения функциональности между ПЕ в большей мере рутинный, чем творческий, и пригоден для автоматизированной продержки.

Поскольку артефакт создается на основе модели подзадач, то средства редактирования должны обеспечивать возможность сопоставить подзадачам из функциональной модели программные единицы с теми же названиями и возможность редактирования названий.

Метод: Основной задаче сопоставляется корневой агент; «композитным» задачам (узлам модели с ненулевой полустепенью исхода) могут быть сопоставлены агенты с двумя блоками, конечным подзадачам (в «листовых» узлах) – обрабатывающие ПЕ. Для оптимизации числа ПЕ «листовые» подзадачи могут быть реализованы в ПЕ, соответствующих верхним уровням по усмотрению проектировщика.

Пример. Сеть ПЕ (блоков агентов) может быть такой:

постановка и объяснение диагноза [корневой агент]
формирование объяснения [имя агента-сотрудника]
{группирующий [тип агента]
Оценить гипотезу здоров [имя агента-сотрудника]
{группирующий [тип агента]
Оценка гипотезы «признак в норме» [имя агента-сотрудника]
обрабат [тип агента]
сделать Вывод о здоровье [имя агента-сотрудника]
обрабат [тип агента]}
Оценка гипотезы имеется заболевание [имя агента-сотрудника]
{группирующий [тип агента]
Оценка гипотезы «признаки-не-КК в норме»
[имя агента-сотрудника]
Оценка гипотезы «КК- признак соотв заболеванию»
[имя агента-сотрудника]
{группирующий [тип агента]
Проверка варианта КП [имя агента-сотрудника]
обрабат [тип агента]
Вывод о том, соответствует ли КК- признак заболеванию
[имя агента-сотрудника]
обрабат [тип агента]}
сделать Вывод о заболевании [имя агента-сотрудника]
обрабат [тип агента]}}.

Здесь ПЕ – «Оценка гипотезы здоров» для выполнения своей работы должна проверить нормальность всех относящихся к делу признаков, т.е. обратиться к ПЕ «Оценка гипотезы "признак в норме"» для каждого из признаков. После того, как все признаки будут оценены, можно будет делать общий вывод о том, здоров ли пациент.

Модель связей по управлению. Выбор способа взаимодействия агентов в процессе решения задачи является «более творческой» стадией по сравнению с построением сети агентов.

Для спецификации взаимодействия выбирается один из повторно используемых шаблонов сообщений, или проектируется новый шаблон. При выборе политики раздельного проектирования интерфейса и логики приложения модель может содержать дополнительных агентов, обеспечивающих интерфейс с пользователем [4].

Метод построения модели связей по управлению по модели подзадач и сети агентов таков. Корневой агент запускается инициализирующим сообщением; агенты, сопоставленные подзадачам в некотором подчинении, реагируют на сообщение-задание, в случае «цикличного» подчинения такое сообщение имеет параметр, схожий с переменной цикла (ссылку на элемент конечного множества, хранимого в некоторой ИСС). Агент для подзадачи в «простом» подчинении, как правило, формирует ответное сообщение «обратная связь», а в «цикличном» подчинении – сообщение-связь с циклом.

Наиболее удобным для восприятия человеком является представление взаимосвязей между ПЕ в виде графа, однако для поддержки единообразного хранения этой модели, методов доступа к ее элементам в процессе автоматизации ее построения и использования требуется и вариант ИСС. Структура сети «управляющий граф» может быть такой:

```
управляющий граф агентного решателя {
  Имя приложения [строка]
   ~ set вершина-блокАгента {
              имя блока [строка]
              имя агента [строка]
              шаблон для запуска блока [$ шаблон]};
   ~ set дуга-сообщение {
              Вершина-отправитель [->имя блока]
              шаблон для следующего [$ шаблон];
              управляющая метка [строка]
              Вершина-получатель [строка] }}
```

Ожидаемая автоматизация поддержки разработчиков на этом этапе такова. Целесообразна генерация дуг с шаблонами инициализирующим и сообщение-задание, а для «обратных» дуг – выбор из двух вариантов: сообщений «обратная связь», сообщение-связь с циклом. При этом каждой для редактирования свойств каждой дуги необходима возможность смены шаблона сообщения.

Пример. Промежуточный шаг процесса построения сети агентов и структуры их взаимодействия может выглядеть так.

Подзадаче «Оценка гипотезы здоров» сопоставляется так называемой <группирующий агент> «Оценить гипотезу здоров», это означает, что агент должен получить ответы на свои поручения, чтобы подытожить результат их выполнения (прежде, чем делать вывод, ему надо убедиться в том, что уже проверены все случаи). Он имеет более одного блока: блок реагирования на сообщение-задание (с параметрами «оценить норму», «ссылка на ИБ», …) и блок реагирования на

сообщение-обратная связь или связь-с-циклом (с параметрами «ссылка на ИБ», «проверенный случай/признак», оценка нормы для случая/признака). Последний блок помимо контроля полноты проверки также выполняет подзадачу «сделать заключение о признании здоровым», подчиненную рассматриваемой подзадаче.

Дальнейшее проектное моделирование. Обусловленный предшествующими моделями выбор архитектуры решателя (рис. 4), в свою очередь повлияет на дальнейшие модели, ведущие к реализации. Дальнейшее проектирование связано с добавлением в архитектурную модель информационных связей ПЕ, в частности через операции над хранимыми информационными ресурсами (ИР). Корневой агент, например, может проверить непустоту входных данных (в хранимых ИР), следовательно, требуется специфицировать ПЕ – операцию для выполнения такой проверки используемого ИР (в виде ИСС). Обрабатывающие агенты нуждаются в доступе к ИР, поэтому при архитектурном планировании должны быть предусмотрены все необходимые для этого операции. Агенты, производящие заключения и фиксирующие их в выходном ИР, нуждаются в операциях модификации ИСС. «Группирующий» агент должен убедиться в том, что проверены все элементы множества, ему требуется доступ к элементам этого конечного множества (хранится как поддерево сем сети). Поэтому ожидается, что для обеспечения качества каждого решателя ИПС требуется построить и согласовать такие его модели, как проект информационных связей агентов, спецификации и проекты интерфейсов каждого агента и проекты каждого хранимого ИР.

Достаточность представляемых (проектируемых) наборов операций над ресурсами проверяется их присутствием в спецификациях множества агентов. Для решателей задач, работающих с ИСС, проект (архитектура) хранимого ИР – описание его структуры и набор спецификаций операций, достаточных для манипулирования ими. Если имеет место повторное использование и АТД ранее разработан для некоторого ИР, то не исключена необходимость новой операции доступа к ИР такого типа, тогда формируются спецификации на разработку. Проектные решения по информационным связям ПЕ могут быть представлены как расширение одной из построенных моделей, например, сети агентов. Расширение состоит в указании для каждого обрабатывающего агента-сотрудника множества обрабатываемых ИР и необходимых для такой обработки операций.

Структура сети для представления такой модели:

Расширенная сеть ПЕ{
имя корневого агента [строка]
{~set хранимое данное {~set имя операции; }
~ set сотрудничество {
имя агента-сотрудника [строка];
тип агента [управл, обрабат, группирующий];
~settт хранимое данное {
~set имя операции; }
~ settт -> сотрудничество}
}}}.

Определив функцию каждой ПЕ и очертив ее интерфейс, проектировщики приступают к разработке (или поручают ответственным за дальнейшую разработку) каждого агента (и множества операций для каждого ИР) повторно-используемых единиц. Спецификации агента и последующие модели, составляющие конфигурацию каждой единицы, также представляются в соответствии с декларативно-агентным подходом.

Структура сети для представления спецификации агента содержит:
имя агента,
описание назначения агента,
~set вход-сообщение {
описание функции агента
шаблон вх сообщения,
список параметров вх сообщения,
~set вых-сообщение {
шаблон сообщения,
список параметров},
~set входной ИР,
~set элемент информации из интерактивного источника,
~set вых ИР}.

Проекты каждого агента из архитектуры целесообразно строить параллельно с моделью связей по управлению.

Пример. Агенту «Оценить гипотезу здоров» (его второму блоку) требуется операция доступа к ИР «объяснение» «Запись в отчет, в норме ли признак». А чтобы сформировать задание для каждого признака, другому агенту надо обратиться к сети «ИБ» с операцией «Взять названия всех признаков». Агент «Оценить гипотезу «признак в норме» должен организовывать проверку всех значений, ему нужна операция «Взять все значения указанного признака», а также операция для обращения к указанной области значений – «взять все значения». Одной из операций доступа к ИР «база знаний о норме» может стать взять область нормальных значений признака («ссылка на БНорм», «признак»): множество значений или диапазон значений. Пример операции доступа к ИР «Объяснение»: добавить значение признака не-в-норме в гипотезу здоров (указанный признак, множество пар <значение, момент наблюдения>).

Литература

1. Белоцерковцева Л.Д. Опыт автоматизации деятельности медперсонала перинатального центра города Сургута / Л.Д. Белоцерковцева // Федеральный справочник «Здравоохранение России». – 2010. – Т. 9. – С. 469-474.
2. Клещев А.С., Грибова В.В., Управление интеллектуальными системами. Известия РАН. Теории и системы управления. 2010. № 6. С. 122-137.
3. Шалфеева Е.А. Семантические модели представления решателя Интеллектуальной системы / Е.А. Шалфеева // Материалы III Междунар. научн.-техн. конф. «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2013) / [отв. ред.: В. В. Голенков]. – Минск : БГУИР, 2013. – С. 257-264.
4. Агентный подход к разработке интеллектуальных интернет-сервисов / [Грибова В.В., Клещев А.С., Крылов Д.А. и др.] // Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT'12». – М. : Физматлит, 2012. – Т. 1. – С. 218-223.
5. Шалфеева Е.А. Разработка модели конфигурации для декларативного программирования управляемых агентных решателей / Шалфеева Е.А. – Владивосток : ИАПУ ДВО РАН, 2012. – 48 с.
6. Pressman R.S. Software Engineering: Practitioner's Approach / R.S. Pressman // Fifth edition. – McGraw-Hill Inc., 2001. – 860 p.
7. Клещев А.С. Системный анализ при автоматизации интеллектуальной профессиональной деятельности / А.С. Клещев, Е.А. Шалфеева // XIII Национальная конференция по искусственному интеллекту с международным участием «КИИ-2012»,16–20 октября 2012 г. – ISBN: 978-5-361-00182-8 Труды конференции, т. 2. – Белгород : Изд-во БГТУ, 2012. – С. 128-135.
8. Клещев А.С., Черняховская М.Ю., Москаленко Ф.М. Модель онтологии предметной области «Медицинская диагностика». Часть 1 : Неформальное описание и определение базовых терминов / А.С. Клещев, М.Ю. Черняховская, Ф.М. Москаленко // Журнал НТИ. – 2005. – № 12. – Серия 2.
9. Москаленко Ф.М. Алгоритм диагностики, основанный на реальной онтологии медицины, для многопроцессорной ЭВМ / Ф.М. Москаленко // Доклад Paco2006. – Издатель : Институт проблем управления им. В.А. Трапезникова РАН, 2006.

10. Mori G. CTTE: support for developing and analyzing task models for interactive system design / G. Mori, F. Paternò, C. Santoro // IEEE Trans. Softw. Eng. – № 28(8). – С. 797-813.
11. Басс Л. Архитектура программного обеспечения на практике / Басс Л., Клементс П., Кацман Р. - 2-е изд. – Л. : Питер, 2006. – 576 с.

Literatura

1. Belotserkovtseva LD Experience in the automation of the medical staff of the perinatal center of the city of Surgut / LD Belotserkovtseva / / Federal handbook "Health of Russia." - 2010. - T. 9. - S. 469-474.
2. Kleshchev AS, Vladimir Gribov, intelligent control systems. Izvestiya. Theory and control systems. 2010. Number 6. S. 122-137.
3. Shalfeeva EA Semantic representation of the model solver Intelligent Systems / EA Shalfeeva / / Proceedings of the III International. nauchn.-technical. Conf. "Open the semantic technology of intelligent systems» (OSTIS-2013) / [Br. Ed.: VV Golenkov]. - Minsk: BSUIR, 2013. - S. 257-264.
4. Agent-based approach to the development of intelligent web-services / [Vladimir Gribov, ticks, AS, DA Krylov and others] / / Proceedings of the Congress on Intelligent Systems and Information Technology «IS & IT'12». - Moscow: Fizmatlit, 2012. - T. 1. - S. 218-223.
5. Shalfeeva EA Development of a model configuration for the managed agent-based declarative programming solvers / EA Shalfeeva - Vladivostok: IACP, 2012. - 48.
6. Pressman R.S. Software Engineering: Practitioner's Approach / RS Pressman // Fifth edition. - McGraw-Hill Inc., 2001. - 860 p.
7. Kleschev AS System analysis in the automation of intellectual profession / AS Ticks, EA Shalfeeva // XIII National Conference on Artificial Intelligence with international participation "KII-2012",16-20 October 2012 - ISBN: 978-5-361-00182-8 Conference Proceedings, Volume 2. - Belgorod State Technological University Publishing House, 2012. - S. 128-135.
8. Kleshchev AS, Chernyakhovskaya MY, Moskalenko FM Model ontology «Medical diagnostics». Part 1: Informal description and definition of basic terms / AS Ticks, M. Chernyakhovskaya, FM Moskalenko // Journal of STI. - 2005. - № 12. - Series 2.
9. Moskalenko FM Diagnostic algorithm, based on a real ontology for multiprocessor computer / FM Moskalenko // Report Paco2006. - Publisher: Institute of Control Sciences. VA Trapeznikova RAS, 2006.
10. Mori G. CTTE: support for developing and analyzing task models for interactive system design / G. Mori, F. Paternò, C. Santoro // IEEE Trans. Softw. Eng. – № 28(8). – С. 797-813.
11. Bass L., Software Architecture in practice / Bass, L., Clements, P., R. Katzman. - 2nd ed. - L.: Peter, 2006. - 576 p.

RESUME

E.A. Shalfeeva

The Method of Construction of Intelligent Problem Solver's Design Representations from the Models of Life Cycle Initial Stages

In given article the approach to development of problem solvers of an intellectual program systems using developed knowledge bases, kept and processed in the form of semantic networks, is offered. According to the concept of modern intellectual systems [2-5], knowledge bases have to be, first, operated by the expert, secondly, be stored in conceptual information resources, and, the third, be accessed by programs. This new approach consists in representation of solver at each development stage in the form of the declarative models. Such models models have to be clear to the developer and are convenient for program access as much as possible to use these results at the subsequent development and maintenance. This approach provides construction of such set of models of each solver which minimizes expenses during its maintenance. Architectural models of a solver are constructed with the regard for conceptual high-level architecture of an whole system. Program units which in the course of decision-making use stored knowledge and handle data, address to the semantic networks storing these data and knowledge.The method of use of some models for construction of the following models and ensuring of program access to all these models are the basis for creation of tools for development of long-living intellectual program systems.

Статья поступила в редакцию 10.04.2013.